Robotic Trajectory Planning Based on CL Data*

Guoqiang Zeng, Chin-Yin Chen, Dishan Huang, Yingdan Zhu

***Abstract*—It is crucial to plan trajectory in robotic finishing with complex surface in manufacturing industry. While traditional robot programming for grinding is still time consuming and not cost-effective generally. This paper focuses on automatic trajectory planning based on the cutter location(CL) data generated in CAD/CAM system. It explores a suitable way to convert the G code which obtained from CL data to robot program directly through Visual Studio, how to automatically extract motion data from G code, generate the KUKA robot paths/programs in PC . In addition, simulation of robot's kinematics models are implemented in MATLAB to verify the robot reachability. An experiment in Unigrahics, MATLAB and real robot illustrates the grinding trajectory composed of CL data is appropriate .**

I. Introduction

Industrial robot is now applied in various industrial field as a kind of flexible automation equipment, for its flexible adaptability of producing conditions and working environment, improving the productive efficiency greatly, and ensuring the quality of product homogeneity. Along with the development of intelligent manufacturing and the industrial 4.0 concept was proposed. Industrial robot's application in the field of mechanical processing will also increase. According to International Federation of Robotics (IFR), sales to the metal and machinery industry reached a new peak level of almost 16,500 units, accounting for a share of 9% of the total supply in 2013[1]. Ranking third only to automotive industry 39% and electrical/electronics industry 20%.

With the rapidly advancing computer technology, it is possible to implement CAD/CAM software to program for robot. Nowadays, millions of users worldwide are using CAD technology to design and model their products for its economically attractive and simply to work. Up to now, the CL data which composed of cutter tip position and tool direction can be directly obtained from the CAM system in the manufacturing industry[2]. The CL data was described as the basic tool path for the multi-axis NC machining. The generation procedure from CC ,CL points and CO angles were also showed in [3]. Ho showed the tool path generation procedure and tool orientation smoothing method in five-axis machining, and presented a cutting error improvement(CEI) method that modify CL data keeping final cutting error within the required tolerance[4]. A new methodology of path generation for rapid prototyping(RP) operations was introduced based on CAD/CAM. In this methodology, the center location source files(CLSF) was utilized to generate the path data for RP operation[5].

*Research supported by STS Project of Chinese Academy of Sciences.

Guoqiang Zeng is with 1Zhejiang Key Laboratory of Robotics and Intelligent Manufacturing Equipment Technology, Ningbo Institute of Materials Technology and Engineering, Chinese Academy of Sciences, Ningbo, Zhejiang, 315201 China and also with the 2School of Mechatronic Engineering and Automation, Shanghai University, Shanghai, 200072 China. (email: zengguoqiang@nimte.ac.cn).

Chin-Yin Chen，Yingdan Zhu(email: y.zhu@nimte.ac.cn) are with the 1Institute of Advanced Manufacturing Technology, Ningbo Institute of Materials Technology and Engineering, Chinese Academy of Sciences, Ningbo, Zhejiang, 315201 China (Corresponding author: Chin-Yin Chen; phone: +86-574-87602663; fax: +86-574-86382329; e-mail: chenchinyin@nimte.ac.cn).

Dishan Huang is with the 2School of Mechatronic Engineering and Automation, Shanghai University, Shanghai, 200072 China. (email: hdishan@shu.edu.cn).

Actually, CAD has been used in the application of robot vision as a new technology in 1980s[6]. Series of study have been done in the area of CAD-based robot path planning and programming from then on. Some research to extend CAD's capabilities in the robotics field were explored over the years.

Neto proposed a CAD-based robot programming system that could directly generate robot programming from a common CAD system in [7]. An adaptive and low-cost robotic coating platform was presented using laser triangulation to extract position and orientation information for robot programming[8]. A CAD-based OLP was explored to extract robot motion sequences from CAD drawing and presented the process of automatic generation of robot path planning and programming[9]. Nagata[10] described a robotic CAM system directly dealt with CL data without using any robot language, making it simple for robot path planning. A robotic polishing system was introduced to take the cutter location which is generated from the post processer on a common CAD in path planning. And a quaternion interpolation(QI) algorithm was proposed between two CL data to realize the smooth motion of end-effector [2]. Zhu[11] presented a dedicated off-line programming system for robotic drilling based on Catia. It is quite clear that researchers have conducted a variety of study in the field of CAD/CAM-based robot programming. However, few of the research showed that the CAD/CAM-based robot programming is really independent of robot cell. Robot's and objects' 3D model are necessary in the simulation of robot motion. It is complicated and time consuming for constructing model before path planning. And the problem of inverse solution of joints along the planning trajectory was not considered, such as multiple solutions.

In this paper, a new method is proposed to automatically convert G code to robot programming through Visual Studio. G code is generated from CL data which composed of cutter tip position and tool direction obtained in CAD/CAM system. In this process, only the workpiece 3D model and its position relative to the base coordinate system of robot are necessary. It is implemented to how to convert milling tool path to robotic grinding trajectory. Setting up KUKA KR60-3's kinematics and inverse kinematic model in MATALB to verify the robot reachability, inverse solutions are also presented based on CL data. The problem of multiple solutions is discussed by means of MATLAB. Finally, a basic experiment is conducted to illustrate the grinding trajectory composed of CL data is suitable in real KUKA robot.

II. ACQUIREMENT OF POSITION AND ORIENTATION

A．Meaning of CL data

Nowadays, all kinds of CAD/CAM systems are widely applied in manufacturing industry, such as Unigrahics, Catia, Pro/Engineer, Solidworks, etc. Each CAM system can generate CL data that consist of position and orientation information through CAD module. If a workpiece is designed in CAD/CAM system and manufactured by CNC tool, the CL data can be regarded as the ideal machining trajectory[9]. So it is crucial to research how to transform the CL data into robot program.

B．Generation of CL data

When CL data are directly applied to industrial robot, we have to ensure that the interval between each CL point is proper or not in the desired trajectory. In grinding processing, accurate position control is dispensable since the end-effcetor has a certain displacement compensation. So processing step distance can be expanded in the range of allowable error to make the robot running more smoothly.

In the grinding processing with KUKA robot, the position and orientation in each point consist of 6 coordinate values. Because of NC milling machining and robotic grinding have a great similarity in the motion mode, it is possible to obtain CL data from milling CNC to generate the motion trajectory of robot grinding processing.

In this paper, CAM system of Unigrahics is chosen to generate CL data. Some parameters are set as follows: processing type[multi-axis mill],cutter diameter[50mm], driven approaching[surface], cutting pattern[zigzag], processing step[9], tool axis[perpendicular to the workpiece].Fig.1 shows the desired trajectory that generated using the main-processor of Unigrahics. And the “GOTO/” statements which consist of position vector and orientation vector is shown in Fig.2,the front three values represent the position information and the queen values are orientation.

Post-processing is a way transferring CLSF into G code that the NC system can be identified. The portion of G code that corresponding to Fig.4 as shown in Fig.3.

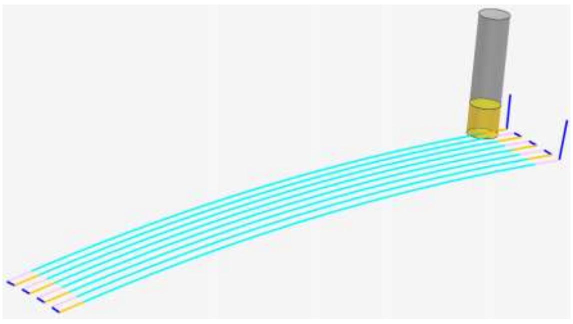
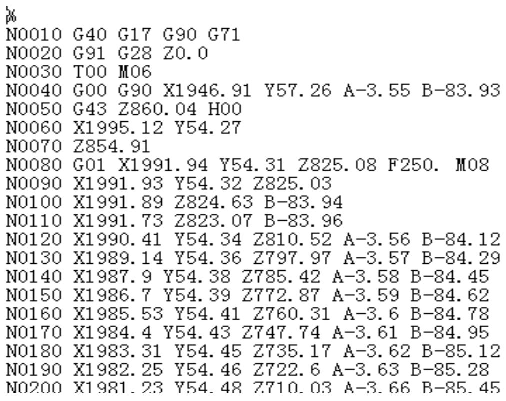


Figure 1. Grinding path consist of CL data

```
GOTO/1991.6846,50.9021,824.6314,-0.9926460
,0.0589766,0.1057147
GOTO/1991.5180,50.9044,823.0633,-0.9926827
,0.0590015,0.1053560
GOTO/1990.2061,50.9243,810.5187,-0.9929714
,0.0591992,0.1024854
GOTO/1988.9307,50.9443,797.9724,-0.9932522
,0.0593926,0.0996119
GOTO/1987.6916,50.9635,785.4220,-0.9935252
,0.0595809,0.0967352
```

Figure 2. Example of CL data written by multi-lined “GOTO/” statements



```
%
N0010 G40 G17 G90 G71
N0020 G91 G28 Z0.0
N0030 T00 M06
N0040 G00 G90 X1946.91 Y57.26 A-3.55 B-83.93 S0 M03
N0050 G43 Z860.04 H00
N0060 X1995.12 Y54.27
N0070 Z854.91
N0080 G01 X1991.94 Y54.31 Z825.08 F250. M08
N0090 X1991.93 Y54.32 Z825.03
N0100 X1991.89 Z824.63 B-83.94
N0110 X1991.73 Z823.07 B-83.96
N0120 X1990.41 Y54.34 Z810.52 A-3.56 B-84.12
N0130 X1989.14 Y54.36 Z797.97 A-3.57 B-84.29
N0140 X1987.9 Y54.38 Z785.42 A-3.58 B-84.45
N0150 X1986.7 Y54.39 Z772.87 A-3.59 B-84.62
N0160 X1985.53 Y54.41 Z760.31 A-3.6 B-84.78
N0170 X1984.4 Y54.43 Z747.74 A-3.61 B-84.95
N0180 X1983.31 Y54.45 Z735.17 A-3.62 B-85.12
N0190 X1982.25 Y54.46 Z722.6 A-3.63 B-85.28
N0200 X1981.23 Y54.48 Z710.03 A-3.66 B-85.45
```

Figure 3. G-code of mill

III. AUTOMATIC CONVERSION OF INSTURCTIONs

A．NC program(G code)

G code is a kind of NC machine instruction for task. NC machine can move straight line, arc and other auxiliary cutting motion by executing the G code. NC program is composed of a series of procedures section, each paragraph usually contains a single step command of processing operation. And the procedure section is usually composed of address character N,G,X,Y,Z,F,S,T,M and the corresponding numerical values.

B．KUKA robot instructions

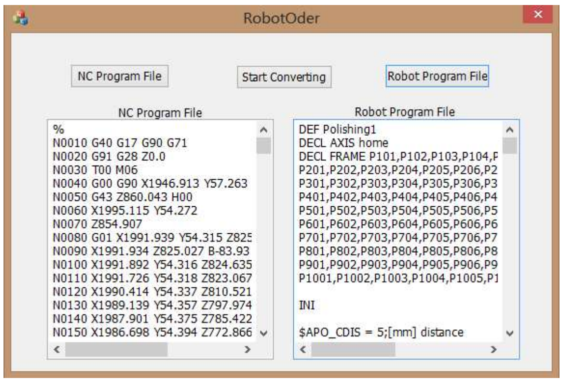
Generally, it is consist of motion instructions, data transmission instructions, I/O instructions, control instructions , operation and auxiliary instructions as a full set of robot instruction system, a part of them are introduced as follows.

Motion instructions are defined in the light of definition of different robot coordinate system or different pulse position to move the specific curve. They are divided into liner moving instruction LIN, arc instruction CIRC, point-to-point instruction PTP, etc.

C．Robot program generation

“DEF program name()”is always appearing at the beginning of the program. “END” indicates the end of program; “DECL” is used to define variables; “INI” contains the standard parameters which the procedures run correct is needed, “INI” which composed of motion instruction and waiting/logic instruction has to be operated firstly. “PTP Home” is usually placed at the beginning and the end, because this is the only known position. “FRAME” defines the position and orientation with 6 numeric data, which is X-, Y-, Z-, A-, B-, C-, respectively. The robot moving speed was decided by "$VEL.CP".

Code conversion module implements transformation from G code to robot control program. On the basis of NC machine's and robot's control characteristics, extracting the relevant information from NC machining program, and then converting the NC program to the robot motion control instruction, saving it as the JOB file at the end. For instance, the "G01" code in NC programming will be converted to "LIN" or "PTP" in robot programming[12].Visual Studio is a suitable tool to accomplish this on account of its visual interface, dialog boxes and related function keys. Fig.4 shows the framework of the automatic conversion process.



```
%
N0010 G40 G17 G90 G71
N0020 G91 G28 Z0.0
N0030 T00 M06
N0040 G00 G90 X1946.913 Y57.263
N0050 G43 Z860.043 H00
N0060 X1995.115 Y54.272
N0070 Z854.907
N0080 G01 X1991.939 Y54.315 Z825
N0090 X1991.934 Z825.027 B-83.93
N0100 X1991.892 Y54.316 Z824.635
N0110 X1991.726 Y54.318 Z823.067
N0120 X1990.414 Y54.337 Z810.521
N0130 X1989.139 Y54.357 Z797.974
N0140 X1987.901 Y54.375 Z785.422
N0150 X1986.698 Y54.394 Z772.866
```

```
DEF Polishing1
DECL AXIS home
DECL FRAME P101,P102,P103,P104,P
P201,P202,P203,P204,P205,P206,P2
P301,P302,P303,P304,P305,P306,P3
P401,P402,P403,P404,P405,P406,P4
P501,P502,P503,P504,P505,P506,P5
P601,P602,P603,P604,P605,P606,P6
P701,P702,P703,P704,P705,P706,P7
P801,P802,P803,P804,P805,P806,P8
P901,P902,P903,P904,P905,P906,P9
P1001,P1002,P1003,P1004,P1005,P1

INI

$APO_CDIS = 5;[mm] distance
```

Figure 4. Automatic conversion process framework

As is well known, when a 6-axis fully articulated industrial robot is applied to grinding, the rotation around the tool axis is irrelevant and requiring only five joints. It is accomplished that the tool axis is aligned with the last joint in this application, so that when confirming the orientation information in CL data, the orientation of TCP can be determined only with roll angle(A) and pitch angle(B) while the yaw angle(C) is flexible[9]. According to the posture between end-effector and workpiece, when the yaw angle(C) is confirmed to 180(°) will be rationalized. Thus, Fig.5 shows the KUKA program that converted from G code which presents in Fig.5, it is composed of position, orientation, moving forms, speed and force parameters, etc.

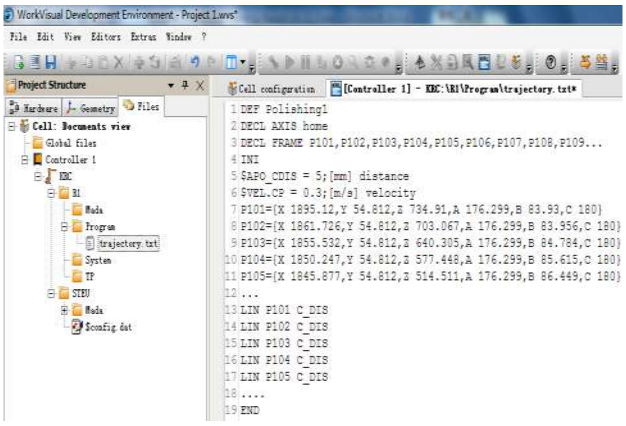
```
DEF Polishing1
DECL AXIS home
DECL FRAME P101,P102,P103,P104,P105,P106,P107,P108,P109,
INI
$APO_CDIS = 5;[mm] distance
$VEL.CP = 0.3;[m/s] velocity
P101={X 1990.41,Y 54.812,Z 810.52,A 176.299,B 84.12,  C 180}
P102={X 1989.14,Y 54.812,Z 797.97,A 176.299,B 83.956,C 180}
P103={X 1987.9,  Y 54.812,Z 785.42,A 176.299,B 84.784,C 180}
P104={X 1986.7,  Y 54.812,Z 772.87,A 176.299,B 85.615,C 180}
P105={X 1985.53,Y 54.812,Z 760.31,A 176.299,B 86.449,C 180}
...
LIN P101 C_DIS
LIN P102 C_DIS
LIN P103 C_DIS
LIN P104 C_DIS
LIN P105 C_DIS
...
END
```

Figure 5. KUKA program after conversion

D. Robot program running

As shown in Fig.5, the program that generated from G code is a KUKA expert programming type. It is also fatal that how the KUKA robot running the generated program. Actually, KUKA robot can only recognize the program when the program files with .src and .dat format at the same time. So here is the way how to connect the KUKA robot with PC.

The WorkVisual software package is the engineering environment for KUKA Robot Controller4(KRC4) controlled robotic cells. It offers transferring projects to the real robot controller. Fig.6 shows the WorkVisual graphical user interface. It is necessary to follow the steps to accomplish the task to run the generated program. Firstly, generating code; then pinning a project; at last, transferring the project to the robot controller.



```
DEF Polishing1
DECL AXIS home
DECL FRAME P101,P102,P103,P104,P105,P106,P107,P108,P109...
INI
$APO_CDIS = 5;[mm] distance
$VEL.CP = 0.3;[m/s] velocity
P101={X 1895.12,Y 54.812,Z 734.91,A 176.299,B 83.93,C 180}
P102={X 1861.726,Y 54.812,Z 703.067,A 176.299,B 83.956,C 180}
P103={X 1855.532,Y 54.812,Z 640.305,A 176.299,B 84.784,C 180}
P104={X 1850.247,Y 54.812,Z 577.448,A 176.299,B 85.615,C 180}
P105={X 1845.877,Y 54.812,Z 514.511,A 176.299,B 86.449,C 180}
...
LIN P101 C_DIS
LIN P102 C_DIS
LIN P103 C_DIS
LIN P104 C_DIS
LIN P105 C_DIS
....
END
```

Figure 6. WorkVisual graphical user interface

IV. KINEMATICS SIMULATION

In order to control the robot's position and orientation better in the machining process, it is quite necessary to set up the kinematics model with mathematical description of robot in the work unit. As we known, any robot could be seen as a series of joints connected together. The Denavit-Hartenberg (D-H) parameters describing the relationship between structures' movement is widely used in the field of robot in the reason of simplicity. KUKA KR60-3 is an industrial robot with 6-axis fully articulated. Fig.7 shows the link coordinate system of KR60-3 in the original condition, and the D-H parameters as shown in TABLE Ⅰ. The axis of Joint4, Joint5 and Joint6 are perpendicular each other and intersected at the same point, and the coordinate system{4} and {5} are coincidence.

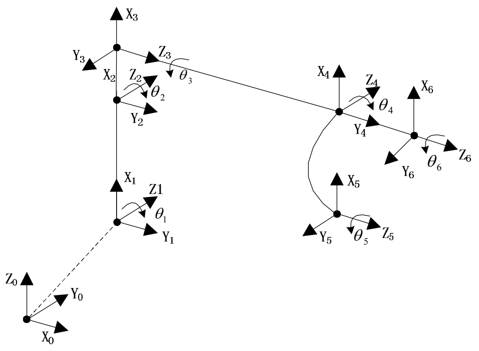


Figure 7. Link coordinate system of KR60-3

Based on the parameters and joint variables of KR60-3 robot, the kinematics model of KR60-3 robot can be set up using function link and serialink of Toolbox in MATALB as shown in Fig.8.

TABLE I. LINK PARAMETERS OF KR60-3

Link i	a_i(mm)	α_i(°)	d_i(mm)	θ_i(°)	Rotary Range
1	350	-90	815	0	±185°
2	850	0	0	-90	+35°/-135°
3	145	-90	0	+90	+158/-120°
4	0	90	820	0	±350°
5	0	-90	0	0	±119°
6	0	0	170	0	±350°

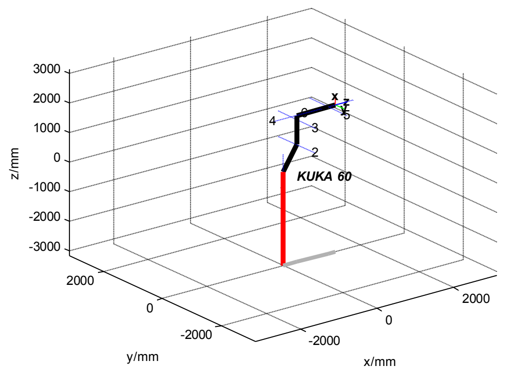


Figure 8. Kinematic model of KUKA60-3

V. INVERSE KINEMATICS SIMULATION BASED ON CL DATA

A. Expression of position and orientation

The position coordinate is directly obtained by CL data. The position of point k is $[X(\text{k}) \quad Y(k) \quad Z(k)]^T$, and the rotation matrix that rotates x, y and z is $\boldsymbol{R}_{x(k)}$, $\boldsymbol{R}_{y(k)}$ and $\boldsymbol{R}_{z(k)}$ as follows:

$$\boldsymbol{R}_{x(k)}=\begin{bmatrix}1 & 0 & 0\\ 0 & \cos A_k & -\sin A_k\\ 0 & \sin A_k & \cos A_k\end{bmatrix} \tag{1}$$

$$\boldsymbol{R}_{y(k)}=\begin{bmatrix}\cos B_k & 0 & \sin B_k\\ 0 & 1 & 0\\ -\sin B_k & 0 & \cos B_k\end{bmatrix} \tag{2}$$

$$\boldsymbol{R}_{z(k)}=\begin{bmatrix}\cos C_k & -\sin C_k & 0\\ \sin C_k & \cos C_k & 0\\ 0 & 0 & 1\end{bmatrix} \tag{3}$$

The rotation matrix of point k is expressed as

$$\boldsymbol{R}_{XYZ}(A,B,C)=\boldsymbol{R}_Z(A)\boldsymbol{R}_Y(B)\boldsymbol{R}_X(C)$$
$$=\begin{bmatrix}cCcB & cCsBsA-sCcA & cCsBcA+sCsA\\ sCcB & sCsBsA+cCcA & sCsBcC-cCsA\\ -sB & cBsA & cBcC\end{bmatrix} \tag{4}$$

($cA=\cos A, sA=\sin A$, the rest follow this rule)

The homogeneous matrix of point k is expressed as

$${}^0_6T(k)=\begin{bmatrix}cC_kcB_k & cC_ksB_ksA_k-sC_kcA_k & cC_ksB_kcA_k+sC_ksA_k & X(k)\\ sC_kcB_k & sC_ksB_ksA_k+cC_kcA_k & sC_ksB_kcC_k-cC_ksA_k & Y(k)\\ -sB_k & cB_ksA_k & cB_kcC_k & Z(k)\\ 0 & 0 & 0 & 1\end{bmatrix} \tag{5}$$

B. Inverse solutions

Generally, there may be as many as 16 kinds of solutions for an industrial robot with six rotary DOFs. However, it is impossible that all the solutions can be reached since each joint can not range 360 degrees. Eight feasible solutions may be obtained for a certain point in robot's workspace.

It is assumed that point P locates at (X 1370.30,Y 54.81,Z 1334.12,A 176.30,B 83.93,C180). Meanwhile the resulting transformation matrix that coordinate system{i} relative to {i-1} as follows:

$$\begin{aligned}{}^{i-1}_{\ \ i}T&=\boldsymbol{R}_Z(\theta_i)\boldsymbol{R}_Z(\theta_i)\boldsymbol{D}_X(a_i)\boldsymbol{R}_X(\alpha_i)\\ &=\boldsymbol{Rot}(z,\theta_i)\boldsymbol{Rot}(z,\theta_i)\boldsymbol{Trans}(a_i,0,0)\boldsymbol{Rot}(x,\alpha_i)\\ &=\begin{bmatrix}c\theta_i & -c\alpha_is\theta_i & s\alpha_is\theta_i & a_ic\theta_i\\ s\theta_i & c\alpha_ic\theta_i & -s\alpha_ic\theta_i & a_is\theta_i\\ 0 & s\alpha_i & c\alpha_i & d_i\\ 0 & 0 & 0 & 1\end{bmatrix}\end{aligned} \tag{6}$$

At this section, importing the parameters in TABLE Ⅰ into Eqs.6, calculating ${}^0_1T$, ${}^1_2T$, ${}^2_3T$, ${}^3_4T$, ${}^4_5T$, ${}^5_6T$ and ${}^0_6T$,then obtaining inverse matrix ${}^1_0T$, ${}^2_1T$, ${}^3_2T$, ${}^4_3T$, ${}^5_4T$ and ${}^6_5T$. In this case, solving eight solutions at the point P as shown in TABLE Ⅱ .

TABLE II. EIGHT SOLUTIONS OF THE CORRESPONDING SPATIAL POINT P

NO.	θ_1(°)	θ_2(°)	θ_3(°)	θ_4(°)	θ_5(°)	θ_6(°)
1	-3.13	-84.59	116.41	164.65	26.53	-166.92
2	-3.13	-84.59	116.41	-15.35	-26.53	13.08
3	-3.13	20.22	-96.35	6.85	82.30	-1.65
4	-3.13	20.22	-96.35	-173.15	-82.30	178.35
5	176.87	-173.17	34.78	-170.85	48.08	-6.86
6	176.87	-173.17	34.78	9.15	-48.08	173.14
7	176.87	-148.68	-14.72	-162.86	23.65	-16.50
8	176.87	-148.68	-14.72	17.14	-23.65	163.50

The criterion of selecting solution is always mutative for the problem of multiple solutions. However, the reasonable choice should be "the shortest route" which means each joints moves the minimum angle. In the case of without obstacles, it will be obtained "the shortest route" solutions in joint space by using algorithm, while there are several ways to decide "the shortest route" solutions. For example, a typical robot is composed of three big links and three small links, and the posture links are next to the end-effector. Thence, it has to weighted when calculating "the shortest route" to trend to move the small links and not to move the big links, that is following the principle "more moving big links, and less little ones" [13].

C. Inverse kinematics simulation

Fig.9 shows the method establishing the inverse kinematics model of KR60-3 based on MATLAB Robotics Toolbox.

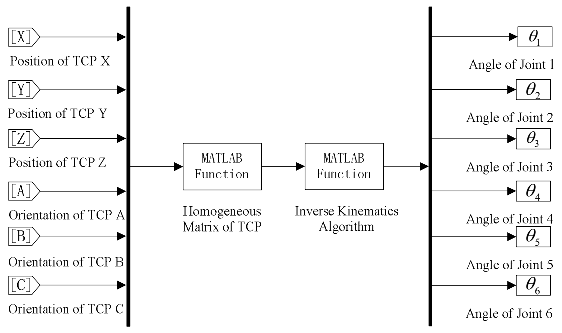


Figure 9. The establishment method of inverse kinematics model

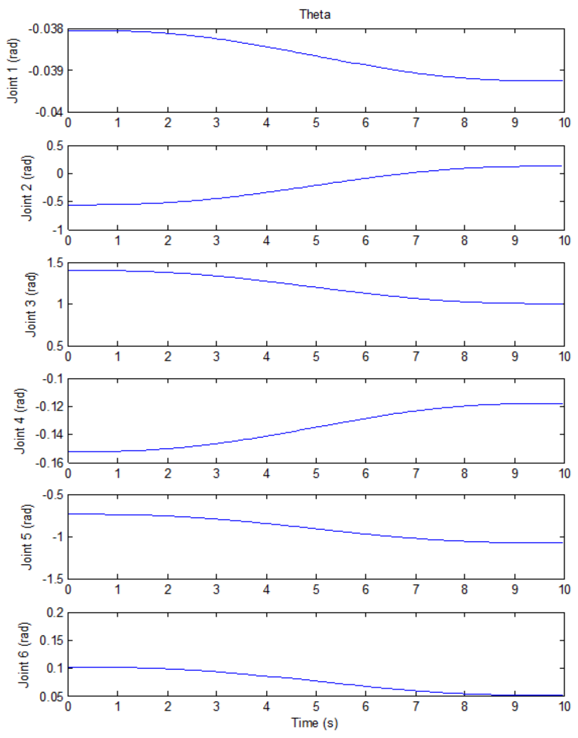
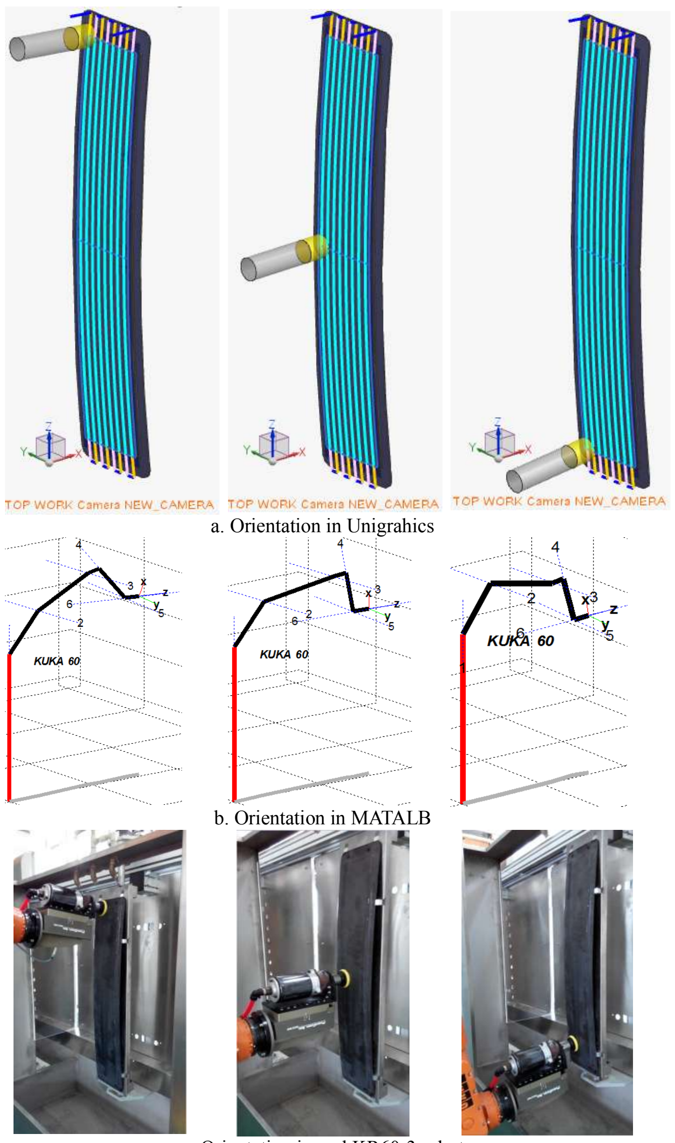


Figure 10. Simulation curve of each joint variable

An actual grinding trajectory was implemented to verify the inverse kinematics solutions of KR60-3. At the beginning, the value of each joint is(-2.18°,31.97°, 79.95°, -8.72°, -42.21°, 5.86°), the end is(-2.25°, 7.4°, 57.21°, -6.75°, -61.71°, 2.89°), and the sample time is 10 seconds. Each joint's angle simulating curve changing is quite gentle, and there is no mutations within the scope of permit, which is shown in Fig.10.

VI. EXPERIMENT IMPLEMENT AND RESULTS

An experiment of verification was conducted using Unigrahics, MATLAB and KUKA KR60-3 robot. Three points on the continuous path were chosen to compare the position and orientation, they were（X 1991.89,Y 54.32,Z 824.64）、（X 1967.58,Y 54.86,Z 367.34）、（X 1991.73,Y 55.17,Z -88.38）, respectively. Since the grinding path is a curve line, the orientation of each point was changed slightly. While the three points which selected were extraordinary that corresponding to the workpiece's top, middle and bottom, respectively. In this case, the orientation of three special points in Unigrahics, MATLAB and KUKU KR60-3 as shown in Fig.11, they are coincident indeed. It is means that the CL data obtained from Unigrahics can be used in real KUKA robot correctly.



c. Orientation in real KR60-3 robot

Figure 11. Position and orientation of three points in three modes

VII. CONCLUTION

This paper proposed an off-line programming method based on CL data obtained from CAD/CAM system. It used G code as the media, and presented the method to solve position and orientation of points between CL data, then accomplished the connected conversion from CAD/CAM to KUKA Robot Language(KRL) through interface program with Visual Studio. In other words, it was implemented how to convert milling tool path to robotic grinding path. Last but not the least, the grinding control program could be automatic generated in PC. In this way, it can improve the programming efficiency and reduce the cost to buy commercial OLP software.

In addition, kinematics models of KR60-3 were established. A method to solve inverse solutions based on CL data and link parameters was listed. Finally, an experiment was conducted to illustrate the grinding trajectory composed of CL data is suitable in the real KUKA robot.

Our future work will concentrate on two issues: (a) accomplish the position/force hybrid control in grinding process; (b) grinding process parameters optimization for workpiece.

AKNOWLEDGEMENT

The authors would like to thank the members of the Zhejiang Key Laboratory of Robotics and Intelligent Manufacturing Equipment Technology, CNITECH who helped to conduct the simulation on MATLAB and operate KUKA robot .

REFERENCES

[1] "Executive Summary: World Robotics 2014 Industrial Robots," International Federation of Robotics,11-24, January 2015.

[2] F. Y. Lin, and L. Tian-Sheng, "Development of a robot system for complex surfaces polishing based on CL data," *International Journal of Advanced Manufacturing Technology*, vol. 26, no. 9-10, pp. 1132-1137, 2005.

[3] Y. R. Hwang, "Cutting Error Analysis for Table-Tilting Type Four-Axis NC Machines," *International Journal of Advanced Manufacturing Technology*, vol. 16, no. 4, pp. 265-270, 2000.

[4] M.-C.HO, Y. Hwang, and C. Hu, "Five-axis tool orientation smoothing using quaternion interpolation algorithm," *International Journal of Machine Tools & Manufacture*, vol. 43, no. 3, pp. 1259-1267, 2003.

[5] E. Cerit, and I. Lazoglu, "A CAM-based path generation method for rapid prototyping applications," *International Journal of Advanced Manufacturing Technology*, vol. 56, no. 1-4, pp. 319-327, 2011.

[6] B. Bhanu, "CAD-based robot vision," *Computer*, vol. 20, no. 8, pp. 13-16, 1987.

[7] P. Neto, N. Mendes, R. Araújo, J. Norberto Pires, and A. Paulo Moreira, "High-level robot programming based on CAD: dealing with unpredictable environments," *Industrial Robot: An International Journal*, vol. 39, no. 3, pp. 294-303, 2012.

[8] M. Ferreira, A. P. Moreira, and P. Neto, "A low-cost laser scanning solution for flexible robotic cells: spray coating," *International Journal of Advanced Manufacturing Technology*, vol. 58, no. 9-12, pp. págs. 1031-1042, 2012.

[9] P. Neto, and N. Mendes, "Direct off-line robot programming via a common CAD package," *Robotics & Autonomous Systems*, vol. 61, no. 8, pp. 896–910, 2013.

[10] F. Nagata, S. Yoshitake, A. Otsuka, K. Watanabe, and M. K. Habib, "Development of CAM system based on industrial robotic servo controller without using robot language," *Robotics and Computer-Integrated Manufacturing*, vol. 29, no. 2, pp. 454-462, 2013.

[11] W. Zhu, and W. Qu, "An off-line programming system for robotic drilling in aerospace manufacturing," *International Journal of Advanced Manufacturing Technology*, vol. 68, no. 9-12, pp. págs. 2535-2545, 2013.

[12] X.M. Lei, "Industrail robot machining technology research based on CAD/CAM", Master dissertation, Lanzhou University of Technology, Lanzhou, Gansu Province, China, 2012.

[13] S.D.Sun.,*Basis of Industrial Robot Technology*, Xi'an: Northwestern Polytechnical University Press,2002, pp.16-47.